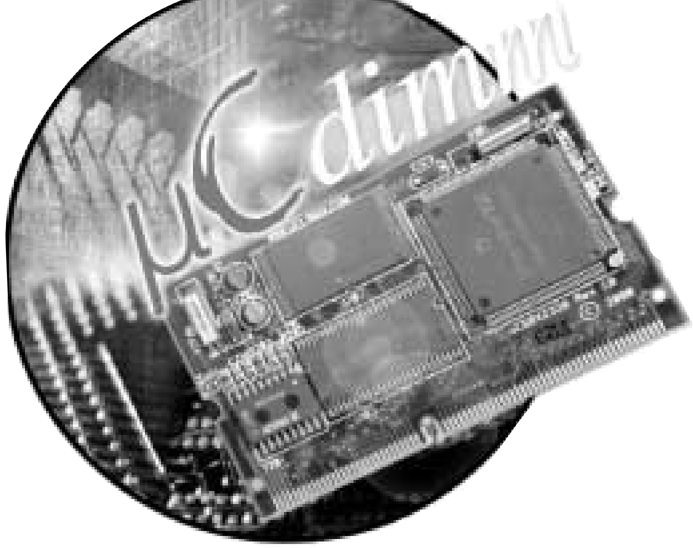


uCdimm ColdFire 5272 Hardware / Firmware Reference Guide

www.arcturusnetworks.com
Arcturus Networks Inc.

Copyright notice

Contact Information

Arcturus Networks, Inc., the Authors and Manufacturers of the uCdimm, uClinux CD-Rom and this manual can be contacted at:

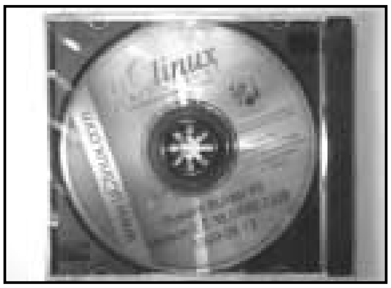
Arcturus Networks Inc.
116 Spadina Ave Suite 100
Phone: +1 416.621.0125
Fax: +1 416.621.0190
URL: www.arcturusnetworks.com

Community use of the uClinux trademark

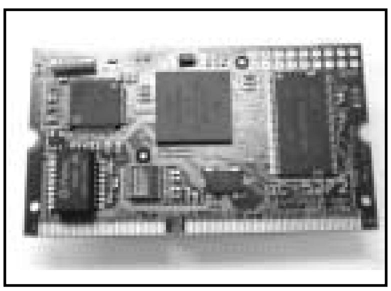
Arcturus Networks encourages the use of uClinux, its trademark name and logo on any and all works as defined under Canadian and US Copyright law as derived works from uClinux subject to conditions of fair and appropriate use. It is the intended spirit that the authors and trademark owners of uClinux be represented and lend their endorsement to derived works, in the support of linux, embedded linux and the Embedded Microcontroller Linux Project (uClinux). Arcturus Networks Inc. and its successors reserve the right to protect on behalf of the community this trademark from any misuse.

Table of Contents

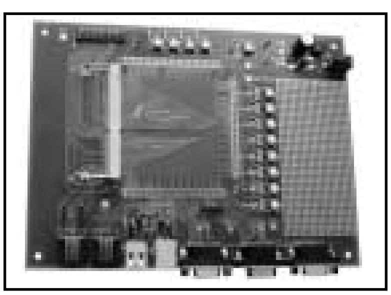
uCdimm Hardware / Software Manual



uClinux System Builder Kit CD-Rom



uCdimm Microcontroller Module

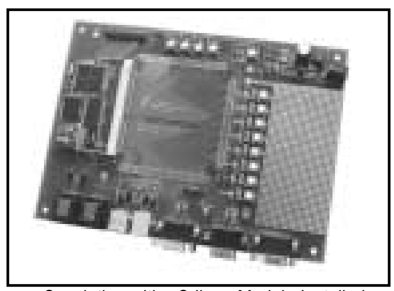


uCevolution Development Platform

List of Tables and Figures

uCdimm Hardware / Software Manual

uCdimm ColdFire Kit Contents



uCevolution with uCdimm Module Installed

1 Introduction

Welcome to the world of embedded Linux

Embedded systems are everywhere. In fact, you touch more embedded systems in your everyday life than you probably realize. They come in many shapes and sizes, from the TV remote control to the “engine computer” in your car. These devices are so prevalent that you likely don’t even realize your interaction with them. They are just there, and they always work (if the battery is charged :-)

One of the interesting things about embedded devices is the wide range of sizes, both in terms of physical size and computing power, in which they are found. Large embedded systems, like the ones found in Automated Teller Machines are likely to be “hardened” PCs. This size of embedded system is normally associated with network connected devices.

What if you could add the kind of functionality normally associated with larger devices to something as common as a home appliance and do it at a low cost? This is the role of uClinux. Designed to run on commodity microcontrollers, uClinux, can be used in place of existing custom code to provide new features that were not originally considered or otherwise possible, resulting in the potential of creating a completely new class of device.

Where there is no existing hardware, the Engineer is free to choose a microcontroller based on the features required and not worry about having to design around a expensive closed source (perhaps even binary only) OS.

The uCdimm hardware takes things one step further. By providing a complete, component level module that can be snapped into a socket to provide these functions, almost any device can be *connected*. The uCdimm can be the core of a system, or it can be a network add on. Core or add on, the uCdimm reduces engineering costs and time to market.

What's on the CD?

The uClinux CD which is included with this kit is a complete development environment for uClinux/ColdFire. To make installation easier, the CD includes binaries and source in RPM format. All the software needed to develop for uClinux from a Linux workstation is included.

System Requirements

The uClinux System requires a Linux host machine as the development workstation. Binaries of the compilers, tools and libraries are provided on the uClinux CD for libc6 x86 based machines. Source code is provided for other platforms and can also be used to rebuild the system.

Binary distribution system requirement

- CD-ROM drive
- x86 Linux with RPM (the Red Hat Package Manager) installed
- 16 Meg RAM. 32 Meg Recommended.
- 1 free Serial port
- 10BaseT Ethernet

If the distribution of Linux you are using on your host machine does not come with RPM installed, you can do one of a number of things. The simplest solution is to download the latest RPM source from Red Hat and install it on your machine. Debian Linux provides a program called "alien", which reads packages in many formats. You can use "alien" to install the uClinux binaries. Alternatively you can you can recompile the uClinux system from source, which will also work if your host is not an x86 machine. See http://www.uclinux.com/ for instructions on building the complete uClinux environment from sources.

uCdimm 2

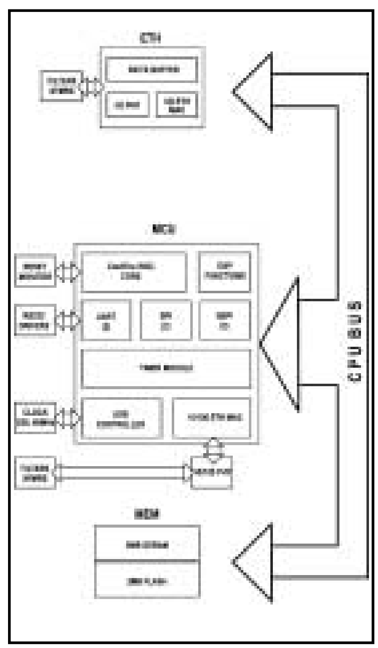
uC5272 Embedded Microcontroller

Linux on a Stick

The uCdimm Coldfire microcontroller module (uC5272) Dual Inline Microcontroller Module is a complete "system on a module" controller, including TCP/IP over dual 10BaseT and 10/100BaseT Ethernet controllers.

Features

- Motorola ColdFire RISC core
- 8Mb SDRAM
- 2Mb FLASH ROM
- 10/100BaseT Ethernet (Incl. Magnetics
- 10baseT Ethernet (Incl. Magnetics)
- 2RS232 serials
- 1 QSPI
- USB controller
- On Board MAC for DSP
- Up to 3 16bit parallel I/O
- 2 PWM outputs
- 4 Hardware Interrupts
- 4 Chip Selects
- 16-bit timer counter module
- Advanced Power Management
- On board uClinux factory installed
- TCP/IP SLIP and PPP
- 1.7" x 2.7" 144 pin soDIMM size
- Cost-effective



uC5272 Architecture

Description

The uCdimm has a highly integrated design which eliminates the need for external hardware normally associated with component level MCUs. Simultaneously, this provides the Engineer functionality which would normally require a far greater “design-in” effort. The uC5272 provides all needed system memory, Ethernet transformers and RS232 line drivers, high speed serial and parallel I/O and the unique uClinux Operating System all on board.

In addition, the uC5272 is supported by a rich development environment including a full ANSI C compliant GNU C tool chain. Example code showing the unique capabilities of the uC5272 for embedded internet appliances and control applications is provided.

The uC5272 is available in an industry standard 144 pin soDIMM form factor ideally suited to both “one off” projects and OEM volume applications. With the uC5272, Engineers can design a completely *connected* device with a small PCB footprint in a matter of hours from concept to prototype.

Basic Architecture

The uC5272 consists of 3 functional blocks: the CS8900 Ethernet Controller, the System Memory and the MCU core. These three functional blocks form a highly integrated component. No external components are required to support a complete system with the exception of a regulated 3.3Volt power supply.

The ColdFire 5272 Core

The on board ColdFire is based on a 32 bit RISC architecture which based on the ColdFire V2 core and provides bus control logic (including SDRAM controller). Also included is a 10/100 BaseT Ethernet controller, a Multiply and Accumulate (MAC) unit for DSP, UARTs, SPI (high speed serial), Timer/PWM and Parallel I/O.

External Memory

The uCdimm module includes 2Mb of FLASH ROM and 8Mb of SDRAM. configured in 16bit wide memories. SDRAM refresh is handled transparently by the MCU core. Both the SDRAM and FLASH ROM go into low power shutdown automatically when idle. The ethernet controller is mapped off the ColdFire 5272 `CSB0` chip select, and is typically located at `0x10000300`.

Memory Map

Address Range		Function
0x00000000	0x0001FFFF	Bootloader System RAM
0x00020000	0x007EFFFF	Operating System RAM
0x007F0000	0x007FFFFC	Bootloader Stack RAM
0x30000000	0x30000FFF	CS8900A Ethernet Controller
0x10C00000	0x10C0FFFF	Bootloader FLASH Image
0x10C20000	0x10DFFFFF	Operating System FLASH
0x10000000	0x100017FF	ColdFire Peripheral Configuration (MBAR)

I/O Memory

I/O memory is mapped into the CPU's main memory space at 2 locations.
The I/O peripherals and system control registers appear at `0x10000000` and extend to `0x100017FF`. The ColdFire 5272 provides an emulation chip select in this address range as well, however, this feature is not available on the uC5272 module. For more detailed information about the registers, see the later sections of this manual or the Motorola MCF 5272 Users' Guide.

FLASH ROM

The FLASH ROM device on the uC5272 is an AMD 29LV, 29DL or compatible series 3.3Volt FLASH device. The exact FLASH device used on any given uC5272 module depends on market conditions. The Bootloader included with the module provides the necessary system calls to manage its FLASH device. The Reset Monitor provides additional under voltage write/erase protection to avoid accidental data corruption and guarantees the FLASH device returns to a known state at "power on" or "reset".

Program and erase of the FLASH device is handled by system calls into the bootloader. There is no need to program directly to the hardware and this is discouraged. See the bootloader chapter for details on the FLASH services offered.

SDRAM

The SDRAM device on the uC5272 provides 8Mb of SDRAM. Refresh is handled by the ColdFire MCU and is configured by the bootloader or the OS kernel at run time.

After reset, the bootstrap runtime configures the SDRAM device and copies the bootloader code from FLASH to the first 128k of SDRAM. From this point, the bootloader no longer relies on the FLASH, allowing it to be erased and reprogrammed. This first 128k block of SDRAM also contains the global environment variables, default fault handlers and the debug stubs. It is therefore important not to corrupt this block of memory.

Ethernet Controllers

The uC5272 contains an on board 10BaseT CrystalLan CS8900A and an on-chip 10/100 BaseT ethernet controller on-the ColdFire MCU. All additional circuitry required to implement a Ethernet support is included on the uC5272 module with no requirements for external components (such as magnetics). Driver code for the CS8900A and the on chip Ethernet controller must be provided by the Operating System running on the module. No support is provided in the Bootloader with the exception of the Arcturus Networks provided IEEE assigned OUI (MAC addresses) which are contained in environment variable `HWADDR0` which contains the MAC address for the on-chip 10/100 BaseT Ethernet controller and `HWADDR1` which contains the MAC address for the on-board 10BaseT Ethernet controller The assigned MAC addresses of the module can also be obtained through the bootloader system call `gethwaddr(0)`and `gethwaddr(1)` corresponding to each controller.

RS232

The uC5272 provides two, 5 wire (RXD, TXD,RTS,CTS and GND) RS232 ports capable of running at up to 5Mbps. RS232 line drivers are integrated; no external components are required. The bootloader will initialize RS232 port 0 at 9600,8,N,1 and uses it as the System Console at reset if the Global Environment Variable `CONSOLE` is set to `ttyS0` (this is the default).

The RS232 line drivers will automatically sense if a powered on RS232 device is connected by checking the voltage on the RXD pin. If no RS232 device is detected, the line drivers automatically go into low power shutdown.

Take note:	**The uC5272's serial ports cannot be used to provide power to peripheral devices such as a mouse.**

QSPI

The uC5272 provides a four wire (Dout, Din, CLK and CS) Motorola QSPI (Queued) serial connection with one chip select . Direct connection to a large range of peripherals, including D/A and A/D converters, UARTS, DSPs and other SPI slaves are supported without additional components.

PWM and Hardware Timer

Two 8 bit PWM modules are provided.
Two identical 16 bit Hardware Timers with 60nS resolution are also provided.

The on board PWM's are capable of telephone quality sound generation. External filtering and transducer driver are required. Other uses include LCD contrast or backlight intensity or other low accuracy D/A conversion functions.

The general purpose timers can be used for input transition event capture or output toggle mode. A prescaler is provided.

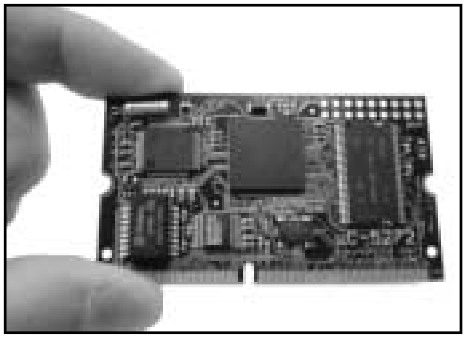
Hardware Multiply / Accumulate (MAC) Unit

The on-board MAC unit provides hardware support for signal processing in a variety of applications including digital audio. The MAC unit and is capable of both 16 bit and 32 bit word lengths.

Universal Serial Bus (USB) Controller

The USB device controller supports data communications to a host provider, typically a PC device. 127 additional devices can be added to the USB chain. The USB controller uses a 4 wire connector model and supports fast (12Mbps) and slow (1.5Mbps) speeds. USB driver code is not provided with current uClinux distributions.

Signal Descriptions



Left: a photo of the uCdimm 5272 (uC5272)

Power and Ground signals

Signal Function	DIMM Bus Pin Number	DIMM Bus Description
Voltage +3.3v	9, 89, 99, 125, 143	VDD
Ground 0v	10, 28, 90, 100, 126, 144	GND

The uC5272 requires a single regulated 3.3volt power supply. The pins provided for VDD (+3.3v) and GND (0v) are listed above. See the section Electrical Specifications for more details.

Reset

Signal Function	DIMM Bus Pin Number	DIMM Bus Description
Reset	98	/RESET

The /RESET signal is an active low, internally pulled up signal. For most applications, this signal can be left unconnected. The on board Reset Monitor will ensure that the module will properly reset after power is applied. The Reset Monitor will also hold the module in reset if the power supply drops below 2.9 volts. When this signal is asserted, the CPU resets, the FLASH device returns to normal READ mode, the ethernet controller is disabled (but remains powered up) and most I/O pins go into High Impedance state. The uC5272 comes out of reset 200mS after both /RESET is deasserted and the power supply stabilizes above 2.9 volts (see the MAX3225 datasheet).

TI/O PWM Outputs

PWMO1 and PWMO2, the pulse width modulator outputs (see the MCF5272 user's manual section 18).

TI/O, the timer counter input capture and output signal. (see the MCF5272 user's manual, section 15). .

Coldfire Description	Signal Function	DIMM Bus Pin Number	DIMM Bus Description
PWM_OUT0	Pulse Width Modulator	51	PWM01
PWM_OUT1	Pulse Width Modulator	52	PWM02
_TA	Clock Timer TIN / TOUT	75	CLKC

QSPI

ColdFire Description	Signal Function	DIMM Bus Pin Number	DIMM Bus Description
QSPI_Dout	Data Out	43	STXD
QSPI_Din	Data In	45	SRXD
QSPI_CLK	Clock	49	SCLK
SPI_CS0	Chip Select	47	SS1

The QSPI interface consists of 4 pins and is multiplexed with functions on RESET

`QSPI_Dout` the (STxD) Data pin is the serial data output from the QSPI (see the MCF5272 users' manual, section 14).

`QSPI_Din` the (SRxD) Data pin, is the serial data input to the QSPI (see the MCF5272 users' manual, section 14).

`QSPI_CLK` (SCLK) Data pin, is the clock output from the QSPI (see the MCF5272 users' manual, section 14).

SPI_CS0, the chip select (SS1) pin when used, select an external device as the source or destination of serial data. Three additional QSPIs can be supported through the chip selects 1..3. (see the MCF5272 users' manual, section 14).

RS232 Port A / ttyS0

Signal Function	DIMM Bus Pin Number	DIMM Bus Description
Received Data	17	RXD1
Transmitted Data	11	TXD1
Request to Send	15	RTS1
Clear to Send	23	CTS1

The RS232 Port A / ttyS0 consists of 4 dedicated pins RXD1, TXD1, RTS1 and CTS1 pins. These pins have internal RS232 line transceivers.

RXD1, the RS232 port A receive pin. This pin is connected through an RS232 line receiver (see the MAX3225 datasheet) to the UART receiver (see the MCF5272 users manual, section 16).

TXD1, the RS232 port A transmit pin. This pin is connected through an RS232 line driver (see the MAX3225 datasheet) to the UART transmitter (see the MCF5272 users manual, section 16).

RTS1 is the RS232 port A Request to Send signal. This pin is connected through an RS232 line driver (see the MAX3225 datasheet) to the UART RTS (see the MCF5272 users manual, section 16).

CTS1 is the RS232 port A Clear to Send signal. This pin is connected through an RS232 line driver (see the MAX3225 datasheet) to the UART CTS (see the MCF5272 users manual, section 16).

RS232 Port B / ttyS1

Signal Function	DIMM Bus Pin Number	DIMM Bus Description
Received Data	18	RXD2
Transmitted Data	12	TXD2
Request to Send	16	RTS2
Clear to Send	24	CTS2

The RS232 Port B / ttyS1 consists of 4 dedicated pins RXD2, TXD2, RTS2 and CTS2 pins. These pins have internal RS232 line transceivers.

RSRXD2, the RS232 port B receive pin. This pin is connected through an RS232 line receiver (see the MAX3225 datasheet) to the UART receiver (see the MCF5272 users manual, section 16).

RSTXD2, the RS232 port B transmit pin. This pin is connected through an RS232 line driver (see the MAX3225 datasheet) to the UART transmitter (see the MCF5272 users manual, section 16).

RSRTS1 is the RS232 port B Request to Send signal. This pin is connected through an RS232 line driver (see the MAX3225 datasheet) to the UART RTS (see the MCF5272 users manual, section 16).

RSCTS2 is the RS232 port B Clear to Send signal. This pin is connected through an RS232 line driver (see the MAX3225 datasheet) to the UART CTS (see the MCF5272 users manual, section 16).

10/100 BaseT Ethernet Interface

The 10/100 BaseT interface provides RX and TX pairs for direct connection to a 10/100Mb/s twisted pair ethernet LAN. Support for this interface is provided by the ColdFire 5272 MCU.

Signal Function	DIMM Bus Pin Number	RJ45 Connector - A Pin Number	DIMM Bus Description
Negative Receive Data	7	6	ERXD1-
Positive Receive Data	5	3	ERXD1+
Negative Transmit Data	3	2	ETXD1-
Positive Transmit Data	1	1	ETXD1+

10BaseT Ethernet Interface

The 10BaseT interface provides RX and TX pairs for direct connection to a 10Mb/s twisted pair ethernet LAN. Support for this interface is provided by the CS8900 Ethernet controller.

Signal Function	DIMM Bus Pin Number	RJ45 Connector - B Pin Number	DIMM Bus Description
Negative Receive Data	8	6	ERXD2-
Positive Receive Data	6	3	ERXD2+
Negative Transmit Data	4	2	ETXD2-
Positive Transmit Data	2	1	ETXD2+

General Purpose I/O Port A

ColdFire Description	Signal Function	DIMM Bus Pin Number	DIMM Bus Description
PC0	General Purpose I/O	53	PA0
PC1	General Purpose I/O	54	PA1
PC2	General Purpose I/O	55	PA2
PC3	General Purpose I/O	56	PA3
PC4	General Purpose I/O	57	PA4
PC5	General Purpose I/O	58	PA5
PC6	General Purpose I/O	59	PA6
PC7	General Purpose I/O	60	PA7

General Purpose I/O Port B

ColdFire Description	Signal Function	DIMM Bus Pin Number	DIMM Bus Description
PC8	General Purpose I/O	63	PB0
PC9	General Purpose I/O	64	PB1
PC10	General Purpose I/O	65	PB2
PC11	General Purpose I/O	66	PB3
PC12	General Purpose I/O	67	PB4
PC13	General Purpose I/O	68	PB5
PC14	General Purpose I/O	69	PB6
PC15	General Purpose I/O	70	PB7

General Purpose I/O Port C

ColdFire Description	Signal Function	DIMM Bus Pin Number	DIMM Bus Description
PA0	General Purpose I/O	71	PC0
PA1	General Purpose I/O	72	PC1
PA2	General Purpose I/O	73	PC2
PA3	General Purpose I/O	74	PC3

Universal Serial Bus

ColdFire Description	Signal Function	DIMM Bus Pin Number	DIMM Bus Description
USB_D-	USB-A Data-	14	UD1-
USB_D+	USB-A Data+	20	UD1+

Hardware Interrupts

ColdFire Description	Signal Function	DIMM Bus Pin Number	DIMM Bus Description
/INT2	Interrupt	77	IRQ0
/INT3	Interrupt	78	IRQ1
/INT4	Interrupt	79	IRQ2
/INT6	Interrupt	80	IRQ3

CPU BUS

The CPU bus is available on the uCdimm module connector please see the uCdimm schematic in appendix C for details.

Electrical Characteristics

The Electrical Characteristics of the uC5272 are a composite of its components. When looking at individual datasheets, be aware of the other components on the module.

Maximum Ratings

Ratings	Symbol	Value	Unit
Supply Voltage	V_{pp}	-0.5 to +4.0	Volt
Input Voltage (except RS232, ETH)	V_{in}	-0.5 to +7.0	Volt
Operating Temperature Range	T_a	0 to 70	C
Storage Temp Range	T_{stg}	-55 to 150	C

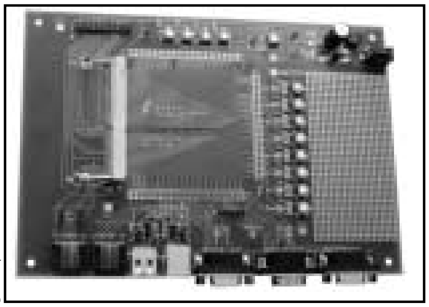
DC Operating Characteristics at 3.3V

Ratings	Symbol	Min	Max	Unit
Supply	I_{pp}	-	110	mA
Standby	I_{pps}	-	1.5	mA
Input High	V_{ih}	2	-	Volt
Input Low	V_{il}	-	0.8	Volt
Output High	V_{oh}	2.4	-	Volt
Output Low	V_{ol}	-	0.4	Volt

uCevolution

Development Platform

3



Above: a photo of the uCevolution development platform

uCevolution Common I/O bus

The uCevolution utilizes the Arcturus Networks uCdimm I/O bus common to all uCdimm products. This offers you the ability to swap uCdimm modules and thus to develop on multiple platforms and processor cores. It should be pointed out that not all processor modules support all the features available in the common I/O bus. Thus you need to familiarize yourself with common I/O bus features supported on the specific uCdimm module you are using.

Health Indicator

The health indicator located between the transmit and receive indicators of the RS232-A(DTE) and RS232-B(DCE) ports is designed to inform you of the status of your uCdimm module. The Heath indicator will change state during context switches while running uClinux.

General Purpose I/O

The uCevolution is equipped with 8 touch-switches and LED's connected to general purpose I/O.

Standard Connectors

A variety of connectors are available on the uCevolution including: 1x RS232-A (DTE & DCE), 1x RS232-B(DCE),1x USB-A, 1x USB-B, 2x Ethernet and LCD. Again, note functionality on the uCevolution board is dependent on support from the uCdimm module

Arcturus Networks uCevolution Bus Description

Pin #	Signal	I/O	Description	Pin #	Signal	I/O	Description
1	ETXD1+	O	Pos. Transmitted Data	2	ETXD2+	O	Pos. Transmitted Data
3	ETXD1-	O	Neg. Transmitted Data	4	ETXD2-	O	Neg. Transmitted Data
5	ERXD1+	I	Pos. Received Data	6	ERXD2+	I	Pos. Received Data / DSL
7	ERXD1-	I	Neg. Received Data	8	ERXD2-	I	Neg. Received Data / DSL
9	VDD	P	VDD	10	GND	P	Ground
11	TXD1	O	Transmitted Data	12	TXD2	O	Transmitted Data
13	DTR1	O	Data Terminal Ready	14	UD1-	I/O	USB-A (Female) DATA-
15	RTS1	O	Request To Send	16	RTS2	O	Request To Send
17	RXD1	I	Receive Data	18	RXD2	I	Receive Data
19	CD1	I	Carrier Detect	20	UD1+	I/O	USB-A (Female) DATA+
21	DSR1	I	Data Set Ready	22	UD2-	I/O	USB-B (Male) DATA-
23	CTS1	I	Clear To Send	24	CTS2	I	Clear To Send
25	RI1	I	Ring Indicator	26	UD2+	I/O	USB-B (Male) DATA+
27	SGND1	P	Signal Ground	28	SGND2	P	Signal Ground
29	ETH 01 LINK	I/O	Ethernet-1 Link LED	30	ETH 02 LINK	I/O	Ethernet-2 Link LED
31	ETH 01 DATA	I/O	Ethernet-1 Data LED	32	ETH 02 DATA	I/O	Ethernet-2 Data LED
33	ANALOG1	I/O	Analog	34	ANALOG2	I/O	Analog
35	ANALOG3	I/O	Analog	36	ANALOG4	I/O	Analog
37	AVDD	P	Analog VDD	38	AGND	P	Analog Ground
39	TCK	I	JTAG	40	TDO	O	JTAG
41	TDI	I	JTAG	42	TMS	I	JTAG
43	MOSI1	I/O	Master Output Slave Input / STxD	44	MOSI2	I/O	Master Output Slave Input / STxD
45	MISO1	I/O	Master Input Slave Output / SRxD	46	MISO2	I/O	Master Input Slave Output / SRxD
47	/SS1	I/O	Slave Select / SCHS	48	/SS2	I/O	Slave Select / SCHS
49	SPICLK1	I/O	Serial Peripheral Interface Clock / SCLK	50	SPICLK2	I/O	Serial Peripheral Interface Clock / SCLK
51	PWMO1	O	Pulse Width Modulation	52	PWMO2	O	Pulse Width Modulation
53	PA0	I/O	Port A Bit 0	54	PA1	I/O	Port A Bit 1
55	PA2	I/O	Port A Bit 2	56	PA3	I/O	Port A Bit 3
57	PA4	I/O	Port A Bit 4	58	PA5	I/O	Port A Bit 5
59	PA6	I/O	Port A Bit 6	60	PA7	I/O	Port A Bit 7
61	CLKA	O	Clock A	62	CLKB	O	Clock B / Health Indicator
63	PB0	I/O	Port B Bit 0	64	PB1	I/O	Port B Bit 1
65	PB2	I/O	Port B Bit 2	66	PB3	I/O	Port B Bit 3
67	PB4	I/O	Port B Bit 4	68	PB5	I/O	Port B Bit 5
69	PB6	I/O	Port B Bit 6	70	PB7	I/O	Port B Bit 7
71	PC0	I/O	Port C Bit	72	PC1	I/O	Port C Bit 1
73	PC2	I/O	Port C Bit 2	74	PC3	I/O	Port C Bit 3
75	CLKC	O	Clock C	76	CLKO	O	Processor Clock Output
77	/IRQ0	I/O	Interrupt	78	/IRQ1	I/O	Interrupt
79	/IRQ2	I/O	Interrupt	80	/IRQ3	I/O	Interrupt
81	/CS0	O	Chip Select	82	/CS1	O	Chip Select
83	/CS2	O	Chip Select	84	/CS3	O	Chip Select
85	DMACK	I	DMA Acknowledge	86	DMARQ	O	DMA Request
87	/CAS0	O	Column Address Strobe	88	/CAS1	O	Column Address Strobe
89	VDD	P	VDD	90	GND	P	Ground
91	/RAS0	O	Row Address Strobe	92	/RAS1	O	Row Address Strobe
93	/WE	O	Write Enable	94	/DWE	O	DRAM Write Enable
95	/LWE	O	Lower Write Enable	96	/UWE	O	Upper Write Enable
97	/OE	O	Output Enable	98	/MR	I	Master Reset
99	VDD	P	VDD	100	GND	P	Ground
101	A0	O	Address BUS	102	A1	O	Address BUS
103	A2	O	Address BUS	104	A3	O	Address BUS
105	A4	O	Address BUS	106	A5	O	Address BUS
107	A6	O	Address BUS	108	A7	O	Address BUS
109	A8	O	Address BUS	110	A9	O	Address BUS
111	A10	O	Address BUS	112	A11	O	Address BUS
113	A12	O	Address BUS	114	A13	O	Address BUS
115	A14	O	Address BUS	116	A15	O	Address BUS
117	A16	O	Address BUS	118	A17	O	Address BUS
119	A18	O	Address BUS	120	A19	O	Address BUS
121	A20	O	Address BUS	122	A21	O	Address BUS
123	A22	O	Address BUS	124	A23	O	Address BUS
125	VDD	P	VDD	126	GND	P	Ground
127	D0	I/O	Data BUS	128	D1	I/O	Data BUS
129	D2	I/O	Data BUS	130	D3	I/O	Data BUS
131	D4	I/O	Data BUS	132	D5	I/O	Data BUS
133	D6	I/O	Data BUS	134	D7	I/O	Data BUS
135	D8	I/O	Data BUS	136	D9	I/O	Data BUS
137	D10	I/O	Data BUS	138	D11	I/O	Data BUS
139	D12	I/O	Data BUS	140	D13	I/O	Data BUS
141	D14	I/O	Data BUS	142	D15	I/O	Data BUS
143	VDD	P	VDD	144	GND	P	Ground

Take note: Shading indicates features unavailable on the uC5272

uClinux Installation 4

Installing the uClinux System

Installing uClinux

The uCdimm ColdFire CD-ROM contains all of the tools necessary to develop embedded applications with uClinux. Provided are binaries and source code for the following components of the Kit

- uClinux-2.4.17.kernel
- system libraries and compilers
- filesystem tools
- system utilities and example code

Installing the uClinux tools

In order to build code for uClinux, you need to install the uClinux/m68k cross compilers (also called the toolchain). Although you can build the uClinux toolchain from source, we recommend using the pre built binaries.

To install the tools ...

```
# mount -t iso9660 /dev/cdrom /cdrom
# cd /cdrom
# make
```

Take note:	You will need to be user root to install this correctly.

Take note:	The uClinux package, the kernel and userland sources are very large and can take some time to install.

This procedure installs the uClinux toolchain into `/usr/local.` This procedure also installs the distribution archive into `/opt/uClinux`.

```
$ ls /opt/uClinux
```

to create a working environment as a non-privileged (not root) user

```
$ buildenv
```

This procedure unpacks the distribution archive into the uClinux-coldfire subdirectory

```
$ ls uClinux-coldfire
   COPYING        Makefile bin      lib         uClibc
   Documentation  README   config   linux-2.4.x user
   MAINTAINERS    SOURCE   freeswan tools       vendors
```

Adding User Applications to the uClinux Distribution

There are several steps that need to be considered when adding a user-written application to the uClinux configuration system. Entries must be added to three files, and an appropriate `Makefile` must exist in the user application source directory. Given a base distribution directory for example: `/home/user/uClinux-coldfire.`

The three files to edit are

```
user/Makefile
config/Configure.help
config/config.in
```

```
user/Makefile
```

Add a line to the file like:

```
dir_$(CONFIG_USER_TEST_TESTAPP)+= test
```

This adds the directory 'test' to the list of directories to be built.

```
config/Configure.help
```

This file contains the text which is presented on request during the config.
Add a block like:

```
CONFIG_USER_TEST_TESTAPP
This program is a test of adding a userland application
to the distribution.
```

The text must be indented two spaces, and there must be no empty lines. Lines should be no longer than 70 characters long.

```
config/config.in:
```

Add a line in the appropriate menu section (i.e. in the program group you want your application to show up in during the 'make config' process for example, 'misc'

```
bool 'test application'    CONFIG_USER_TEST_TESTAPP
```

The `config_user_ symbol` is structured by application directory followed by program name. In this example the application directory is 'test' and the program name is 'testapp'. This structure allows for directories which contain multiple executables.

Next, there needs to be a proper `user/test/Makefile`. which should follow the following template:

```
   EXEC = testapp
all: $(EXEC)
$(EXEC):
   $(CC) $(LDFLAGS) -o $@ $@.c $(LDLIBS)

romfs:
   $(ROMFSINST) /bin/$(EXEC)

clean:
   rm -f $(EXEC) *.elf *.gdb *.[oa] *~ core
```

If more than one executable is built in the 'test' directory, as above, then the Makefile would reflect that as follows:

```
   EXECS = testapp testapp2
all: $(EXECS)

$(EXECS):
   $(CC) $(LDFLAGS) -o $@ $@.c $(LDLIBS)

romfs:
   $(ROMFSINST) -e CONFIG_USER_TEST_TESTAPP      /bin/testapp
   $(ROMFSINST) -e CONFIG_USER_TEST_TESTAPP2  /bin/testapp2
```

More complex makefiles are of course possible. The reader is encouraged to browse the user tree for examples.

After this is complete a standard `make xconfig; make dep; make` should build the application and install it in `romfs` and hence in the target system `image.bin`.

A closer look at the build engine

We need a method of controlling the behavior of what is built and included in the `image.bin` The two files that control the generation of the `image.bin` are `Makefile` and `deftemplate.sh`.

`Makefile` contains the dependency logic that controls the build process. It assembles all the pieces that make up the uClinux distribution into your working directory and builds them.

Take note:	**You don't have to specify a template; we didn't in the above example. The `Makefile` supplied the default template that was installed with uClinux, `deftemplate.sh`. It's best to start with `deftemplate.sh` and edit it to suit your project. This is a normal shell script, which can execute arbitrary commands.**

A template shell script is used by `Makefile` to populate `romdisk` with the binaries you choose for your project. You can specify the template script you wish to use on the command line (see example below). Do not place any additional spaces; this will confuse the shell.

```
$/testing > make TEMPLATE=mytemplate.sh
```

Contents of the `romdisk` directory: uClinux root filesystem

The `romdisk` directory is a staging area for the root filesystem to be built into the ROMfs image. You can put anything you want in the `romdisk` directory, as long as it fits in your FLASH ROM. Your uClinux ROM filesystem will be a copy of this directory tree (read only of course) The build engine does not remove binaries or other files from the `romdisk` staging area. If you take something out of your template script, you'll need to remove it from the `romdisk` directory before building your new FLASH image, otherwise the old copy will still be included.

The standard uClinux root filesystem has a similar structure to a traditional Linux or UNIX system, except it's a lot smaller. In fact, the image generated using the default template contains only around 140 normal files, directories or device nodes. This can be trimmed down considerably, with the minimum required being a device node for the serial port and a shell. Included with the uClinux distribution at time of this writing are over 50 utility and example programs. Most of these are not required.

Setting up your Workstation to Connect to the Target

You'll need to make sure your development machine has the correct permissions to allow access to the serial port connected to the target. Also, you'll need to make sure the NFS server is configured to export your working environment.

Serial ports under Linux are accessed as the files `/dev/ttyS`*`n`* where *`n`* is a number starting at `0`; that is, `/dev/ttyS0` is COM1, `/dev/ttyS1` is COM2 and so on. You'll need to set the permissions of the device you'll be using to talk to the target so normal users can access it. You will also need a terminal emulator. We suggest you try `minicom` for your terminal program. `man minicom` will display its user manual. There are other terminal emulators available in linux; however, we find `minicom` to be the easiest to use. Before you can use `minicom` as a normal user, you need to configure it as root for each serial line you will be using. Assuming your target will be connected to the first serial port, `/dev/ttyS0`, the following will set the permissions and let you create a `minicom` configuration for it.

```
# chmod oug+rw /dev/ttyS0
# minicom -o ttyS0
```

When minicom starts up, you will need to set a number of options. Go into "Serial port setup" and set the serial line to `/dev/ttyS0, 9600Bps 8N1` and no hardware or software flow control. You might want to set the Init and Reset strings to empty in "Modem and dialing".

Take note:	**Some Linux distributions don't have the XMODEM protocol program set up properly. Change directory to `/usr/bin` and look for the files `sx`, `rx`, `sz` and `rz`. If `sx` or `rx` are missing, make a symbolic link for it (or them) to the appropriate program(s). eg,** **`ln -sf /usr/bin/rz /usr/bin/rx`** **`ln -sf /usr/bin/sz /usr/bin/sx`** **On certain linux distributions, sx is not an XMODEM protocol program. If this is the case, set minicom to use 'sz -X' for XMODEM protocol.**

You should now be able to use the serial port as a normal user. Start up `minicom ttyS0` (as a normal user) with nothing connected to the port. If all goes well, minicom will come up. If you short pins 2 & 3, what you type will echo to the screen (provided you've properly turned off flow control).

NFS Server Configuration.

The Linux NFS server is different depending on which distribution and vintage of Linux you are running. Refer to the documentation that came with your distribution if you have difficulties; however, here is a brief guide.

> **Take note:** **Some distributions provide graphical tools to help you administer your NFS server. Please consult the documentation provided with your distribution**

As root, edit your NFS server configuration file, `/etc/exports.` Type `man exports` for a complete description of it's contents. Add a line in `/etc/exports` for your working area. Here is a copy of `/etc/exports` from one of our development machines...

```
# See exports(5) for a description.
# This file contains a list of all directories exported to other computers.
# It is used by rpc.nfsd and rpc.mountd.
/home (ro)
```

This exports one single directory tree (everything under `/home`) to anyone with permissions `(ro)` or read only. You may wish to export only specific directories, for instance, to the area where you will be working on uClinux code. You can export as many directory trees as necessary by adding more lines to `/etc/exports`

> **Take note:** **Be careful with exporting filesystems over NFS while connected to the Internet. Although you can restrict access to only trusted machines in `/etc/exports`, there are known security holes in recent Linux NFS servers. These holes have been fixed in newer Linux distributions, but we recommend that you not run an NFS server on an Internet connected machine without a proper firewall.**

For the changes to `/etc/exports` to take affect, you need to restart your NFS server.

```
# /etc/rc.d/init.d/nfsserver stop
# /etc/rc.d/init.d/nfsserver start
```

This will restart the NFS server for most linux distributions.

The uClinux Boot Process

When the uClinux kernel boots up, it executes the program `/sbin/init` This program first executes the shell script `/etc/rc` to finish the process of bringing up the system. Here is the default uClinux `/etc/rc`:

```
#!/bin/sh
#
# system startup.

# set up the hostname
hostname uC5272

# mount & setup the ramdisk
mount -n -t ramfs ramfs /var

mkdir /var/tmp
mkdir /var/log
mkdir /var/run
mkdir /var/lock

# Mount the proc filesystem
mount -t proc proc /proc

# Configure local loopback:
ifconfig lo 127.0.0.1 netmask 255.0.0.0

# Configure internal 10/100 ethernet controller:
ifconfig eth0 inet 192.168.1.200 netmask 255.255.255.0

# Configure CrystalLan 10base-T controller:
# ifconfig eth1 inet 192.168.2.200 netmask 255.255.255.0
# or:
# /bin/dhcpcd eth1 &

# Uncomment this line to add a default gateway
# route add -net 0.0.0.0 netmask 0.0.0.0 gw 192.168.1.100

# Uncomment these lines to automatically mount an nfs filesystem:
# /bin/portmap &
# /bin/mount 192.168.1.100:/tftpboot /mnt
```

Take note: **It is best to start your own programs from `/etc/rc` unless they are programs designed to handle serial line communication, in which case they can be listed in `/etc/inittab`.**

After `/etc/rc` exits, `init` starts up the programs listed in `/etc/inittab`. `init` "hooks up" the standard input, output and error to the terminal line listed and sets the bps rate before the program is `exec()`'d. In this case, the serial line `ttyS0` is set up for 96000bps and the program `/sbin/agetty` is run (`/etc/agetty` presents a login prompt).

```
# inittab for uClinux
# Format:
# ttyline:termcap-entry:getty-command
ttyS0:vt100:/sbin/agetty 9600 ttyS0
ttyS1:vt100:/sbin/agetty 19200 ttyS1
```

Logging In To uClinux

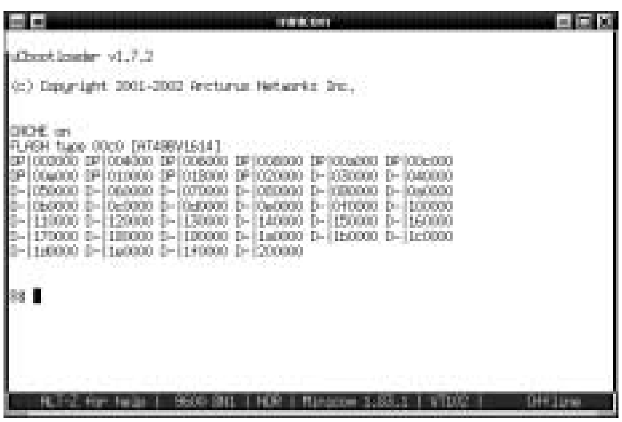
Following the boot process, you will be asked to login to uClinux. The code for the login procedure is located in a file called `login.c`. By default, the username can be any non-empty string, and the password is '**uClinux**' (note the capital 'C').

5 BootLoader

The uCdimm BootLoader: uCbootloader

The uCdimm Bootloader: uCbootloader

The uCbootloader™ serves the same function as a BIOS on a PC, except it resides physically on the same Flash ROM as the uClinux OS and Filesystem and is much more flexible. The Bootloader includes a command line interface for developers and includes the ability to define and use environment variables, enter commands and spawn more copies of the Bootloader shell. uCbootloader initializes hardware to a known state, tests attached devices and relinquishes control to an operating system. In addition, uCbootloader "hooks" exception vectors and attempts to recover from unexpected events during the boot process, providing “device lock out” assurance. Finally, uCbootloader hooks the Trap#2 vector to provide system calls for managing the Flash ROM, reading and writing environment variables and other functions.



The uCdimm defaults into the bootloader shell at system reboot.

The console output shows the type of FLASH device found, and lists the sectors on the device in the following format.

```
DP|004000
|| └004000   This sector ends at offset 0x004000 from the beginning
||           of the FLASH device
|└──────P    This sector is write protected.
└───────D    This sector is Dirty.  It is not completely erased.  All
             sectors are dirty until erased.
```

Bootloader Command Line User Interface

The bootloader provides a command line interface to facilitate interaction with the device from outside an OS. The commands available are described here. Optional parameters are in square brackets [ ].

`sh`
Recursively invokes a new bootloader shell.

`exit`
Exits from the current bootloader shell. If the shell was invoked as a response to an exception, the bootloader attempts to return from the exception and continue execution. If there is nothing to return to, a new shell is spawned automatically.

`help`
Prints a list of commands

`printenv [name]`
Displays the environment variables if permissions allow. If no arguments are specified, printenv prints all environment variables allowed by permissions. If `name` is specified, `printenv` displays only the variable with that name.

`setenv name [value]`
Sets the environment variable with name `name` to the value `value` if `value` is specified. If `value` is not specified, the variable with name `name` is erased. Permissions are checked prior to erasing existing variables, and new variables are created with the current protection mask.

Take note:	**Be Careful not to delete `HWARDDR1` unintentionally**

eraseenv

Erases the FLASH block containing user modifiable environment variables without regard to permissions. When an individual environment variable is erased, it is actually marked as a whiteout entry, and so this command is necessary if the environment space is filled. Data contained in `FACTORY, REVISION, SERIAL` and `HWADDR0` are not user modifiable.

pmask [bsu+-rw]

Sets or displays the current environment variable protection mask. Permissions are individually modifiable for `u` (user) `s` (supervisor) and `b` (bootloader) protection domains. `+` adds permissions, `-` removes them. `r` allows reads of a given variable, `w` allows its contents to be modified or erased in the future.

eg.

`pmask`	Print the current protection mask
`pmask r`	Add read permission to all domains
`pmask -r`	Remove read permission from all domains
`pmask bu+rw`	Add read and write permission to bootloader and user

rx

Receives a binary image through the console port using the XMODEM protocol. The new image is stored in SDRAM stating at address `0x00020000`. After receiving, the image may be burned into FLASH with the `program` command, or executed in RAM with the `goram` command.

program

Erases an area of the FLASH ROM starting at `0x10c20000` (the OS area) and writes the image currently in RAM (received with the `rx` command) into it.

verify

Verifies the image in RAM matches the contents of the FLASH ROM's OS area.

go [-f] [hex_addr]

checks for file type and executes the OS image from FLASH ROM.
[-f] copies the image from FLASH into RAM beginning at 0x00020000 and then attempts to execute the RAM image
[hex_addr] instructs the bootloader to begin executing from a specified location in memory
go supports the use of

`goram`
executes the RAM image.

Take note:	**The commands `go` and `goram` assume that the first four bytes of the image contain the stack pointer address and subsequent four bytes contains the address of the program counter. The bootloader then transfers control over to this location.**

`fast`
change the serial speed to 115200bps. Useful before uploading a large OS image via XMODEM.

`slow`
change the serial speed to 9600bps.

`speed[baudrate]`
change the serial speed [baudrate] = between 9600, and 115200bps
`[baudrate]` = `9600, 19200, 38400, 57600 or 115200`

`md address [endaddress]`
Display a hexdump of the module's memory starting at `address`. If `endaddress` is specified, memory is displayed up to `endaddress`, otherwise 16 bytes will be displayed.

mm <hex_addr> <hex_value>...
Write, a byte at a time, the values listed in `values` into consecutive memory locations starting at `address`.

eg

`mm 00020000 123456` Write the values `0x12`, `0x34` and `0x56` to locations `0x00020000-0x00020002`

Take note:	**Warning! this command, if used improperly, can crash the module.**

`envmm`

Read the environment variable pairs of the form `>address=values...` and write, a byte at a time `values` into consecutive memory locations stating at `address`.

eg

```
setenv >00020000 123456
```

`envmm` Write the values `0x12`, `0x34` and `0x56` to locations `0x00020000-0x00020002`

> **Take note:** **Incorrect values in the `envmm` environment variables can crash the module.**

The uCbootloader is equipped with two additional commands for use with the CRAMFS filesystem

`ls [dir_path]`

Produce a directory listing of the files in cramfs; if filename is specified, list only that file's directory entry. This command returns an error message if the Flash memory does not contain a valid cramfs superblock, or the user-specified file doesn't exist.

> **Take note:** **Due to space limitations, there are none of the usual "ls" command-line options.**

cat <dir_path/file_name>

Display the contents of file filename from the cramfs filesystem. This command will return an error message if; the Flash memory does not contain a valid cramfs superblock, the file does not exist, or the file is actually a directory.

Special Environment Variables

These environment variables affect the operation or bootup sequence of the module.

`FACTORY`	The Arcturus Networks copyright string for the uCdimm design.
`REVISION`	The hardware revision number of the uCdimm
`HWADDR`*n*	The hardware address of network interface *n*. Whereas *n* can either be `0` for the 10/100 Ethernet Controller or `1` for the 10 BaseT.
`SERIAL`	The serial number of this module.
`CACHE`	Enables (on) or disables (off) cache support
`CONSOLE`	Specifies the console device. If `CONSOLE` is `ttyS0` or `yes`, the serial port is initialized to 9600,8,N,1 and used as the console (this is the default). Otherwise, no console if configured. If the bootloader requires operator input, the console is initialized regardless of the value of `CONSOLE`.
`CONSOLE_SPEED`	Specifies the default console speed on bootup, valid speeds include 9600, 19200, 38400, 57600 or 115200. Default is 9600.
KERNEL	Specifies the name of the kernel image to be executed from a CRAMFS filesystems to be run from Start address for
KERNEL_ARGS	Allows for command line passing of arguments to Linux kernels
`AUTOBOOT`	If `AUTOBOOT` is a number, the bootloader will pass control to the OS image after that number of seconds has passed without seeing a character on the console at bootup. If `AUTOBOOT` is `yes` AND `AUTOKEY` is properly set, the module boots into the OS with no delay. All other values for `AUTOBOOT` generate error messages.
`RAMIMAGE`	If `RAMIMAGE` is set to `yes` the bootloader will automatically copy the FLASH image into RAM beginning at 0x00020000 and execute from this location. If the image in Flash is CRAMFS format, the bootloader will perform this operation automatically.
`AUTOKEY`	If set to `iknowmyimageworks`, and if `AUTOBOOT` is set to `yes`, the module will boot directly into the OS FLASH image.
`ENVMM`	If this variable is set to `auto`, the `envmm` command is run at bootup.

Take note: **Warning when `AUTOBOOT` and `AUTOKEY` are set to boot into the OS with no delay, there is NO method to reprogram the FLASH should the OS image prove faulty, leaving the module unbootable.**

Take note: **Warning: test your `ENVMM` instructions first by manually running the `envmm` command and verify the results.**

Writing an OS Image into FLASH ROM from the bootloader

Boot the module to the `B$` prompt. If you have selected an `AUTOBOOT` timeout, press the `<esc>` key on the console terminal within the time specified.

You can change the speed of the console port to 115200bps before uploading if you choose with the `fast` command at the `B$` prompt.

At the `B$` prompt, type `rx` and start the XMODEM upload function of your terminal emulator. Send the OS binary image you wish to program into the module. When complete, type `program`.

The new OS image is now programmed into FLASH ROM.

Bootloader API

The uCbootloader API is accessed through the `TRAP #2` instruction.

The function number is passed in CPU register `D0`
The function arguments are passed in registers `D1-D5`

The return value is passed in register `D0`. If -ve, the system call has failed and the absolute value of `D0` is the error number as enumerated in `booterr.h`

See `bootstd.h` for macros that facilitate making bootloader system calls from C. Some functions listed are deprecated, for testing purposes only or are stubbed out. Only functions recommended for use are listed here.

`void reset(int flags)`

Reset the module. If `flags` has the value `PGM_EXEC_AFTER`, the OS is automatically started after reset.

`void program(mnode_t * chain, int flags)`

Program the new OS image pointed to by `chain` into the OS FLASH area.
`flags` contains the bitwise OR of a combination of
`PGM_ERASE_FIRST`
`PGM_EXEC_AFTER`
`PGM_RESET_AFTER`
`PGM_HALT_AFTER`

If `flags` contains `PGM_ERASE_FIRST`, the memory range to be written will first be erased.

If `flags` contains `PGM_EXEC_AFTER`, the new OS image just written to FLASH will be started. If `flags` contains `PGM_RESET_AFTER`, the module will do a full reset after the image is programmed. if `flags` contains `PGM_HALT_AFTER` the module will go into low power stop after the image is programmed.

If `flags` contains none of `PGM_EXEC_AFTER`, `PGM_RESET_AFTER`, `PGM_HALT_AFTER`, this system call returns after the image is programmed.

unsigned char *gethwaddr(int iface)

Returns a pointer to the MAC address of interface number `iface`. On the uCdimm ColdFire 5272, `iface` should always be `0` or `1`. The MAC address pointed to is in binary format.

char *getserialnum()

Returns a pointer to a string containing the serial number of this module.

char *getbenv(char *var)

Looks up the value of the bootloader environment variable with the name pointed to by `var`. Returns a pointer to a string containing the value of the variable, or `0` if no environment variable with the correct permissions is found. This function checks permissions.

int setbenv(char *pair)

Writes an environment variable pointed to by `pair` in the form `NAME=value`. If `pair` is of the form `NAME`, the environment variable with name `NAME` will be erased. This function checks permissions.

int setpmask(unsigned short pmask)

Sets the protection mask to the value `pmask`. `pmask` is the bitwise OR of the protection permissions listed in `env.h`

char *readenv(int fcn)

Reads environment variables in order. If `fcn` is `0`, `readenv()` returns a pointer to the name of the first stored environment variable. If `fcn` is `1`, `readenv()` returns a pointer to the name of the next stored environment variable. If `readenv()` is `2`, a pointer to the value of the current environment variable is returned. This function checks permissions.

```
int flash_chattr_range( unsigned short *flashprt,
               int start,
               int end,
               char andmask,
               char orfield)
```

Changes the bootloader protection for a range of locations in the FLASH device at base address `flashptr`. `start` and `end` are byte offsets. `andmask` is the bitwise OR of the protection values listed in `flash.h` and is bitwise ANDed with the present value to form the final value. `orfield` is the the bitwise OR of the protection values listed in `flash.h` and is ORed with the present value to form the final value. returns 0 on success.

```
int flash_erase_range( volatile unsigned short *flashptr,
                       int start,
                       int end)
```

If permissions allow, this function erases the range of locations from offset `start` to offset `end` in the FLASH device pointed to by `flashptr`. Returns `0` on success.

```
int flash_write_range( volatile unsigned short *flashptr,
                       mnode_t *chain,
                       int offset)
```

If permissions allow, programs the FLASH device pointed to by `flashptr`, starting at `offset` from the beginning of the device with the image contained in mnode chain `chain`.

Appendix A
uClinux Specific Command Reference

uClinux

uClinux contains many of the commands found in workstation Linux. Some additional commands, or replacements for missing commands are provided. These are documented here.

Command Reference

`expand` File expander

usage: `expand infile outfile`

Expands `infile` compressed with a simple algorithm into `outfile`. The command is used during boot to expand the RAM Disk image.

`flashloader` FLASH image writer

usage: `flashloader imagefile[-d]`

Loads `imagefile` into memory and hands it to the bootloader to write into the OS area of the FLASH. After the OS image is written to FLASH, it is started. `-d` causes debugging information to be displayed

`ramloader` RAM image writer
usage: `ramloader imagefile[-d]`

Loads `imagefile` into memory and hands it to the bootloader to write into the OS area of the RAM. After the OS image is written to RAM, it is started. `-d` causes debugging information to be displayed

`httpd` WWW server

usage: `httpd`
usage: `httpd -i`

Serves files using HTTP. `httpd` is compiled with a hard coded document root, normally set to `/htdocs`. If `httpd` is started with `-i`, it reads the request from standard input and sends the result to standard output (for use with `inetd`).

`ifconfig` Configure a network interface
Configures the specified interface with the given address and netmask.

usage:
```
ifconfig [-a] [-i] [-v] <interface> [[<AF>] <address>]
  [add <address>[/<prefixlen>]]
  [del <address>[/<prefixlen>]]
  [[-]broadcast [<address>]]  [[-]pointopoint [<address>]]
  [netmask <address>] [dstaddr <address>]  [tunnel <address>]
  [outfill <NN>] [keepalive <NN>]
  [hw <HW> <address>]  [metric <NN>]  [mtu <NN>]
  [[-]trailers]  [[-]arp]  [[-]allmulti]
  [multicast]  [[-]promisc]
  [mem_start <NN>]  [io_addr <NN>]  [irq <NN>]  [media <type>]
  [txqueuelen <NN>]
  [[-]dynamic]
  [up|down] ...
```

`route` Defines network routing for an attached network interface

usage: route [-nNvee] [-FC] [<AF>] List kernel routing tables
route [-v] [-FC] {add|del|flush} ... Modify routing table for AF.

`ifattach` Attach a network interface and set up routing

usage: `ifattach [--addr x.x.x.x] [--mask x.x.x.x] \`
`[--net x.x.x.x] [--gw x.x.x.x] [iface]`

Configures the specified interface with the given address, netmask, network address. It then sets up routing based on netmask and network address. If `--gw` is specified, the default route is set to go though the gateway.

If `--addr` is not specified it defaults to `127.0.0.1`
If `--mask` is not specified it defaults to `255.0.0.0`
If `--net` is not specified it defaults to `127.0.0.0`
If `--gw` is not specified, the default route is not set
if `--iface` is not specified, it defaults to `lo`

`inetd` Listen on network ports and spawn programs

usage: `inetd`

`inetd` reads `/etc/inetd.conf` and listens on the ports specified for incoming connections. When a connection arrives, the network connection is hooked up to the standard input, output and error of the specified program (which is spawned). `inetd` reads `/etc/services` and `/etc/protocols` to translate names of protocols and services.

`init` Parent of all processes, init starts the system

usage: none, run by the kernel at boot

`init` is run by the kernel at boot time. It first executes the shell script `/etc/rc` and then enters a loop which keeps the processes listed in `/etc/inittab` running.

`login` Verify a password and `exec()` a shell

usage: `login`
usage: `login -t`

`login` is normally run from `agetty` or `telnetd` when a user tries to login. If `-t` is specified, `login` will first present the login: prompt to ask for a user name, otherwise it assumes `agetty` has asked already and moves right on to password:

> **Take note:** **`login` is a stub implementation. We DO NOT recommend using it in a production environment! It checks the password against the compiled in static string; no other authentication is done. The default password that ships is `uClinux`.**

`reset` Reset the system

usage: `reset`

Appendix B
A Simple Application Note

uCdimm Application Example

The uCdimm makes embedded development simple. Since the uCdimm is a complete system all by itself, you can focus on developing the hardware and firmware that is specific to your project and not waste time reinventing the core. For the most part, the uCdimm is a drop in controller component. It is easy to program; uClinux provides a friendly and complete environment within which to develop application firmware.

Although this example develops as a trivial application, it demonstrates the steps required to develop embedded applications on the uCdimm hardware and the uClinux OS.

Let there be Blinking Lights!

About the simplest thing you can do is blink an LED.

Of course, you need to power the module and connect the serial and Ethernet ports. The uCevolution is an excellent board for prototyping these sorts of things. You can find the schematics and board layout in appendix C of this book.

Before application code work beings, an environment needs to be set up to work in. We need to unpack the development environment and create a FLASH image which things can be added to. We also need a directory to develop our LED blinking code in.

Take note: **We recommend that you put all your projects together into a common directory. The author keeps his uCdimm project code in the directory `/home/dimm/kit.` This lets us export a smaller portion of the workstation via NFS.**

```
$ cd /home/dimm/kit
$ mkdir working
$ cd working
$ buildenv
$ make
$ mkdir test
$ cd test
```

Here is the code to toggle the pin on port C (on the uCdimm) and blinks the LED on PB0 of the uCevolution at 0.5Hz. Use your editor of choice to write this into the file `test.c`

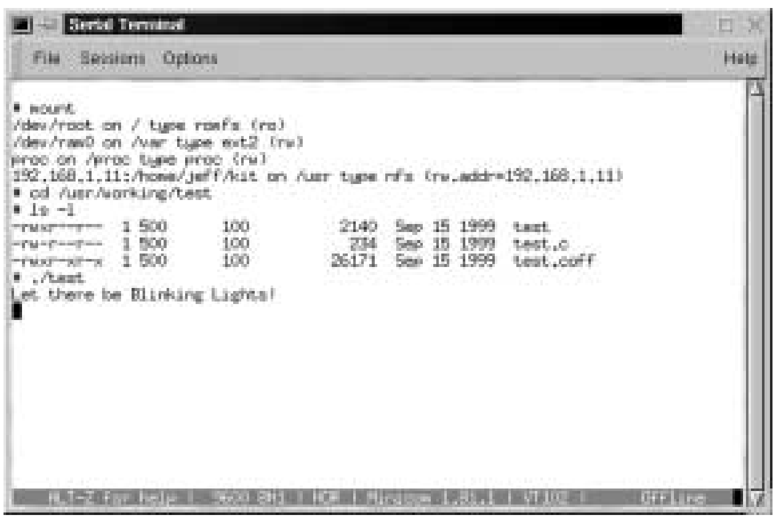
```
#include <unistd.h>
#include <asm/coldfire.h>
#include <asm/m5272sim.h>

int main(int argc, char *argv[]){
        int i = 0;
        int j = 0;
        unsigned char a = 1;
        volatile unsigned short *PCDATA = (volatile unsigned short
      *)(MCF_MBAR + MCFSIM_PCDAT);
        *(volatile unsigned short *)(MCF_MBAR + MCFSIM_PCDDR) =
      0xFF00;
        while(1){
                usleep(500);
                if(j == 0){
                        i++;
                        if(i >= 7 ) j = 1;
                        a <<= 1;
                }
                else{
                        i--;
                        if(i <= 0) j = 0;
                        a >>= 1;
                }
                *PCDATA = (unsigned short)(~a) << 8;
                usleep(500);
        }
}
```

The file `m5272sim.h` contains defines for the ColdFire 5272's on chip registers, and really speeds up development. All we have to do is select the General Purpose I/O function of the port pin we want to use, configure it for output and toggle the corresponding bit in the data register at the right speed. Refer to the section on parallel I/O in the Motorola MCF5272 users Manual for more details on general purpose I/O port D. Now we just need to compile it and we can test the code on the module.

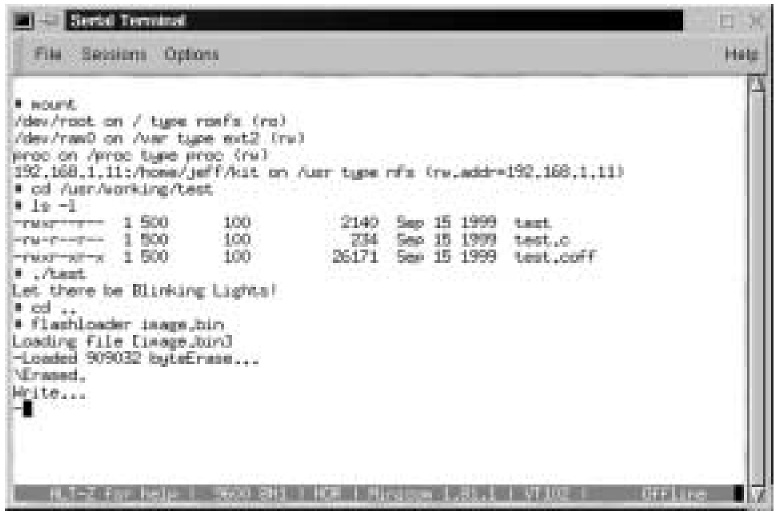
```
m68k-elf-gcc test.c -o test -lc -Wl,-elf2flt -m5307
```

On the module, we can now change into the NFS mounted directory where we compiled the code and test it...

> **Take note:** **It is unnecessary to use assembly code everywhere. With a very good optimizing compiler like GNU CC you will find that very little code will require hand optimization for even time critical IO.**



So far so good, the LED does its thing. Now that we have tested the code, we'd like it to run when the module boots. Refer to Section 4 of this manual for the instructions for adding user applications to the uClinux distribution.



After `flashloader` has successfully written the image to FLASH, the module will perform a warm boot and start blinking the LED as soon as uClinux executes the `/etc/rc` script.

We have provided a very simple example that uses the parallel IO on the uCdimm from uClinux. This example can be expanded further to use any of the onboard peripherals of the uCdimm (eg SPI) provided interrupts are not required. For devices that use interrupts, it is often useful to begin writing code in a user process (as shown above) and then move the code into Kernel later.

Things to watch out for when writing uClinux code

The uClinux memory model is a single flat 32bit address space. All user programs and the kernel share this single address space. As such, there is limited memory protection between programs and between programs and the kernel. Care must be taken not to corrupt the kernel memory or the memory of another process.

Since there is only one 32bit address space shared by all other processes, it is important to be careful with the allocation patterns of applications as the memory space can become fragmented. It is best for a program to allocate as few blocks as possible (preferably only 1) at start up and hand out chunks of that block internally as the program runs.
uClinux does not provide a complete `fork()` implementation. Instead, uClinux provides an implementation of BSD's `vfork()`, a simplified version of `fork()`. With `vfork()`, execution of the parent process is suspended until the child process calls

`exec()` or `exit()`. In most cases the use of `vfork()` will not affect the operation of existing UNIX or Linux programs. See ‘`man fork`’.

However, it is important to realize that both the child process and the parent process share the same stack and global variables. As such, it is important that the child process not return from the function that called `vfork()` or corrupt the parent’s variables before it calls `exec()` or `exit()`. If the child process returns, the parent will find its stack frame corrupted and crash. If the child writes to variables it shares with the parent, the parent will also find them changed.

uClinux implements a large subset of the API implemented in linux. With the exception of the above, writing for uClinux is the same as writing for linux.

Appendix C
Schematics

Appendix D

Licensing, Copyrights & Limitation of Liability

uClinux Distribution

The different parts of the uClinux distribution are licensed under various OpenSource License agreements. Always be sure to examine the code in question for the appropriateness of its license to a given application. Some of the licenses used are reproduced in this section for your reference.

The uClinux Kernel is licensed under the GNU Public License.

Take note:	**Embedded Firmware programs are not considered a derived work of the GNU Public Licensed code that is used in the uClinux distribution and may safely be given a proprietary license. However they are subject to the conditions in the GNU Library General Public License. You must allow your customers to relink your applications with newer versions of the uClinux Standard C Library (libc.a). See the file COPYING.LIB included with libc for details.** **You may not, however, make changes to the uClinux kernel or libraries without releasing your changes as under the same license as used in the existing code. All copyright and licensing notices contained in the source code must be preserved.**

GNU GENERAL PUBLIC LICENSE

Version 2, June 1991

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software—to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Library General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow.

GNU GENERAL PUBLIC LICENSE

TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program", below, refers to any such program or work, and a "work based on the Program" means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term "modification".) Each licensee is addressed as "you".

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

1. You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

a) You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.

b) You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.

c) If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:

a) Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

b) Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

c) Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

4. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

5. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.

6. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

7. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

8. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

9. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

10. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

11. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED,

INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

12. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

Appendix: How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the “copyright” line and a pointer to where the full notice is found.

<one line to give the program’s name and a brief idea of what it does.>
Copyright (C) 19yy <name of author>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 675 Mass Ave, Cambridge, MA 02139, USA.

Also add information on how to contact you by electronic and paper mail.

If the program is interactive, make it output a short notice like this when it starts in an interactive mode:

Gnomovision version 69, Copyright (C) 19yy name of author
Gnomovision comes with ABSOLUTELY NO WARRANTY; for details type `show w’.
This is free software, and you are welcome to redistribute it
under certain conditions; type `show c’ for details.

The hypothetical commands `show w’ and `show c’ should show the appropriate parts of the General Public License. Of course, the commands you use may be called something other than `show w’ and `show c’; they could even be mouse-clicks or menu items—whatever suits your program.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a “copyright disclaimer” for the program, if necessary. Here is a sample; alter the names:

Yoyodyne, Inc., hereby disclaims all copyright interest in the program
`Gnomovision’ (which makes passes at compilers) written by James Hacker.

<signature of Ty Coon>, 1 April 1989
Ty Coon, President of Vice

This General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Library General Public License instead of this License.

The BSD License

The Regents of the University of California Copyright Notice and Disclaimer

Arcturus Networks Inc. - Limitation of Liability, Intended Use.

Arcturus Networks reserves the right to make changes without further notice to any products herein. Arcturus Networks makes no warranty, representation or guarantee regarding the merchantability, suitability or fitness of its products for any particular purpose, nor does Arcturus Networks assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. "Typical" parameters can and do vary in different applications. All operating parameters, including "Typicals" must be validated for each customer application by customer's technical experts. Arcturus Networks does not convey any license under its rights nor the rights of others. Arcturus Networks products are not designed, intended, or authorized for use as components in systems intended for surgical implant into the body, or other applications intended to support or sustain life, or for any other application in which the failure of the Arcturus Networks product could create a situation where personal injury or death may occur. Should Buyer purchase or use Arcturus Networks products for any such unintended or unauthorized application, Buyer shall indemnify and hold Arcturus Networks Inc. and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, and expenses, and reasonable legal fees including, without limitation, court costs arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use, even if such claim alleges that Arcturus Networks was negligent regarding the design or manufacture of the part. i net ready, uCsimm, uClinux, uCbootstrap, uCgardener, uCacademix, uCdimm, uCchip, uCkernel, uCbsd, Geek Kit and GeekCreek and their respective logos are trademarks of Arcturus Networks Inc. Linux is a registered trademark of Linus Torvalds.

Community use of the uClinux trademark

Arcturus Networks Inc. encourages the use of uClinux, its trademark name and logo on any and all works as defined under US Copyright law as derived works from uClinux subject to conditions of fair and appropriate use. It is the intended spirit that the authors and trademark owners of uClinux be represented and lend their endorsement to derived works, in the support of linux, embedded linux and the Embedded Microcontroller Linux Project (uClinux). Arcturus Networks Inc. and its successors reserve the right to protect on behalf of the community this trademark from any misuse..

Copyright notice

Appendix E
References / Suggested Reading

Motorola ColdFireMCF5272 User's Manual
version 2.0
MCF 5272.pdf
by Motorola
(we have provided this pdf manual on the uClinux System Builder Kit CD in the /DataSheets director)

ColdFire Family Programmers Reference Manual
Second Edition
by Motorola
(we have provided this pdf manual on the uClinux CD in the /DataSheets director)

M68000 Family Programmer's Reference Manual
M68000PM/AD REV. 1
by Motorola

Using C on the UNIX System
A Guide to System Programming
by David A. Curry (O'Reilly & Associates, Inc.)
ISBN: 0-937175-23-4

POSIX.4:
Programming for the Real World
by Bill O. Gallmeister (O'Reilly & Associates, Inc.)
ISBN: 1-56592-074-0

Linux Device Drivers
by Alessandro Rubini (O'Reilly & Associates, Inc.)

ISBN: 1-56592-292-1

Linux Kernel Internals
Second Edition
by Beck, Böhme, Dziadzka, Kunitz, Magnus, Verworner (Addison-Wesley)
ISBN: 0-201-33143-8

Using and Porting GNU CC
Version 2.95.2
by Richard M. Stallman (The Free Software Foundation)
ISBN: 1-882114-37-X